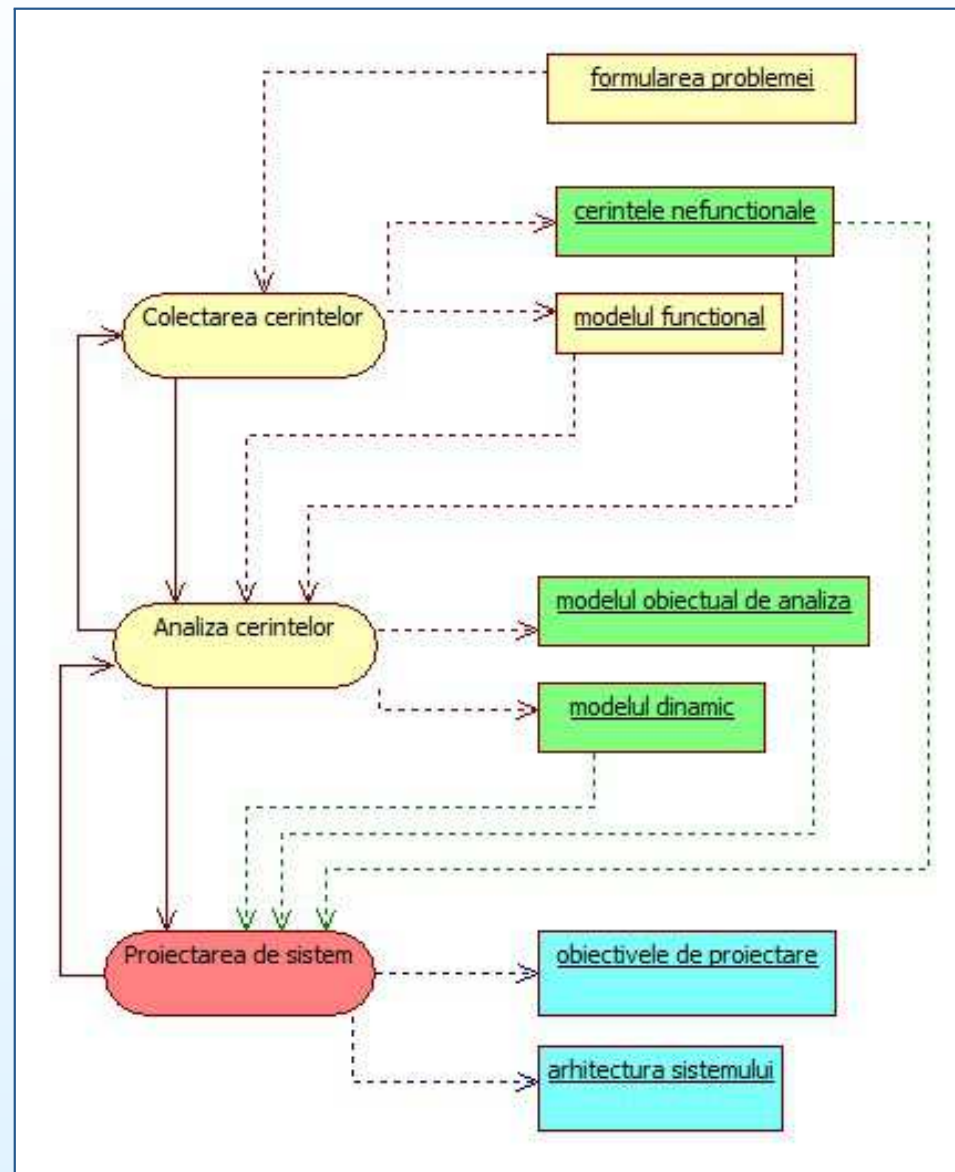


Curs 6
Proiectarea de sistem (II)

Curs bazat pe B. Bruegge and A.H. Dutoit
"Object-Oriented Software Engineering using UML, Patterns, and Java"

Proiectarea de sistem



Proiectarea de sistem (cont.)

- Procesul de transformare a modelului rezultat din ingineria cerințelor într-un model arhitectural al sistemului
- Produse ale proiectării de sistem
 - *Obiectivele de proiectare* (eng. *design goals*)
 - Calități ale sistemului pe care dezvoltatorii trebuie să le optimizeze
 - Derivate din cerințele nefuncționale
 - *Arhitectura sistemului*
 - Subsistemele componente (de dimensiuni mai mici, asignabile unei subechipe de dezvoltare)
 - Responsabilitățile subsistemelor și dependențele între ele
 - Maparea subsistemelor la hardware
 - Strategii de dezvoltare: strategia de gestionare a datelor cu caracter persistent, politicile de control a accesului, fluxul global de control

Proiectarea de sistem (cont.)

- Activități ale proiectării de sistem
 - *Identificarea obiectivelor de proiectare*
 - Identificarea și stabilirea priorităților acelor calități ale sistemului pe care dezvoltatorii trebuie să le optimizeze
 - *Descompunerea inițială a sistemului*
 - Pe baza modelului funcțional și a modelelor de analiză
 - Bazată pe utilizarea unor stiluri arhitecturale standard
 - *Rafinarea descompunerii inițiale pentru a răspunde obiectivelor de proiectare*
 - Rafinarea arhitecturii de la pasul anterior până la îndeplinirea tuturor obiectivelor de proiectare
- Analogie cu proiectarea arhitecturală a unei clădiri
 - Componente: camere vs. subsisteme
 - Interfețe: pereți/uși vs. servicii
 - Reproiectare: mutarea pereților vs. schimbarea subsistemelor/interfețelor

Proiectarea de sistem (cont.)

- Subactivități în rafinarea descompunerii inițiale
 - *Maparea hardware-software*
 - Aspecte urmărite: Care este configurația hardware a sistemului? Cum sunt distribuite funcționalitățile pe noduri? Cum se realizează comunicarea dintre noduri? Ce servicii sunt realizate folosind componente existente? Cum sunt aceste componente încapsulate?
 - Această subactivitate conduce adesea la definirea unor subsisteme adiționale, dedicate transferului de date de la un nod la altul sau gestionării problemelor legate de concurență și fiabilitate
 - *Gestiunea datelor cu caracter persistent*
 - Aspecte urmărite: Care sunt datele cu caracter persistent? Unde ar trebui stocate aceste date? Cum vor fi ele accesate?
 - Această subactivitate conduce adesea la selectarea unui sistem de gestiune a bazelor de date și la identificarea unor subsisteme adiționale dedicate gestiunii datelor cu caracter persistent

Proiectarea de sistem (cont.)

- *Definirea politicilor privind controlul accesului*
 - Aspecte urmărite: Cine are acces la date? La care dintre ele? Se pot schimba dinamic drepturile de acces? Cum este specificat și realizat controlul accesului?
- *Stabilirea fluxului global de control*
 - Aspecte urmărite: Cum sunt secvențiate operațiile în sistem? Este sistemul unul dirijat de evenimente? Poate el gestiona mai mulți utilizatori simultan?
- *Descrierea cazurilor limită (eng. boundary conditions)*
 - Aspecte urmărite: Cum este inițializat și oprit sistemul? Cum sunt gestionate cazurile excepționale?

Exemplu: sistemul *MyTrip*

- *MyTrip* este un sistem care permite planificarea călătoriilor (traseelor aferente) de către șoferi
- Folosind *MyTrip*, un utilizator își poate planifica traseul de efectuat prin intermediul unui calculator personal, prin conectarea la un serviciu Web de planificare (cazul de utilizare *PlanificareTraseu*). Traseul este salvat pe server, în vederea consultării ulterioare. Serviciul de planificare trebuie să suporte diferiți utilizatori
- Ulterior, utilizatorul/șoferul efectuează călătoria cu mașina personală, în timp ce calculatorul de bord îi oferă instrucțiuni, pe baza informațiilor legate de traseu salvate pe server și a poziției curente indicate de sistemul GPS încorporat (cazul de utilizare *EfectuareTraseu*)

MyTrip: cazul de utilizare PlanificareTraseu

Nume	<i>PlanificareTraseu</i>
Flux de evenimente	<ol style="list-style-type: none">1. Șoferul își deschide calculatorul personal și se loghează în serviciul Web de planificare.2. Șoferul introduce o listă de destinații la care dorește să ajungă.3. Folosind o bază de date cu hărți, serviciul de planificare calculează cea mai scurtă variantă de a vizita destinațiile, în ordinea specificată. Rezultatul este dat sub forma unei serii de segmente de drum, ce unesc o serie de intersecții și a unei liste de instrucțiuni.4. Șoferul poate revizui planul, prin inserarea sau ștergerea unor destinații.5. Șoferul salvează traseul sub un nume ales în baza de date a serviciului de planificare, în vederea utilizării ulterioare.

MyTrip: cazul de utilizare EfectuareTraseu

Nume	<i>EfectuareTraseu</i>
Flux de evenimente	<ol style="list-style-type: none">1. Șoferul își pornește mașina personală și se loghează în sistemul de asistență rutieră al calculatorului de bord.2. După logare, șoferul specifică serviciul Web de planificare și numele traseului pe care dorește să îl execute.3. Sistemul de asistență obține lista destinațiilor, instrucțiunilor, segmentelor de drum și intersecțiilor aferente de la serviciul de planificare.4. Pe baza poziției GPS curente, sistemul de asistență oferă șoferului următorul set de instrucțiuni.5. Șoferul ajunge la destinație și oprește sistemul de asistență.

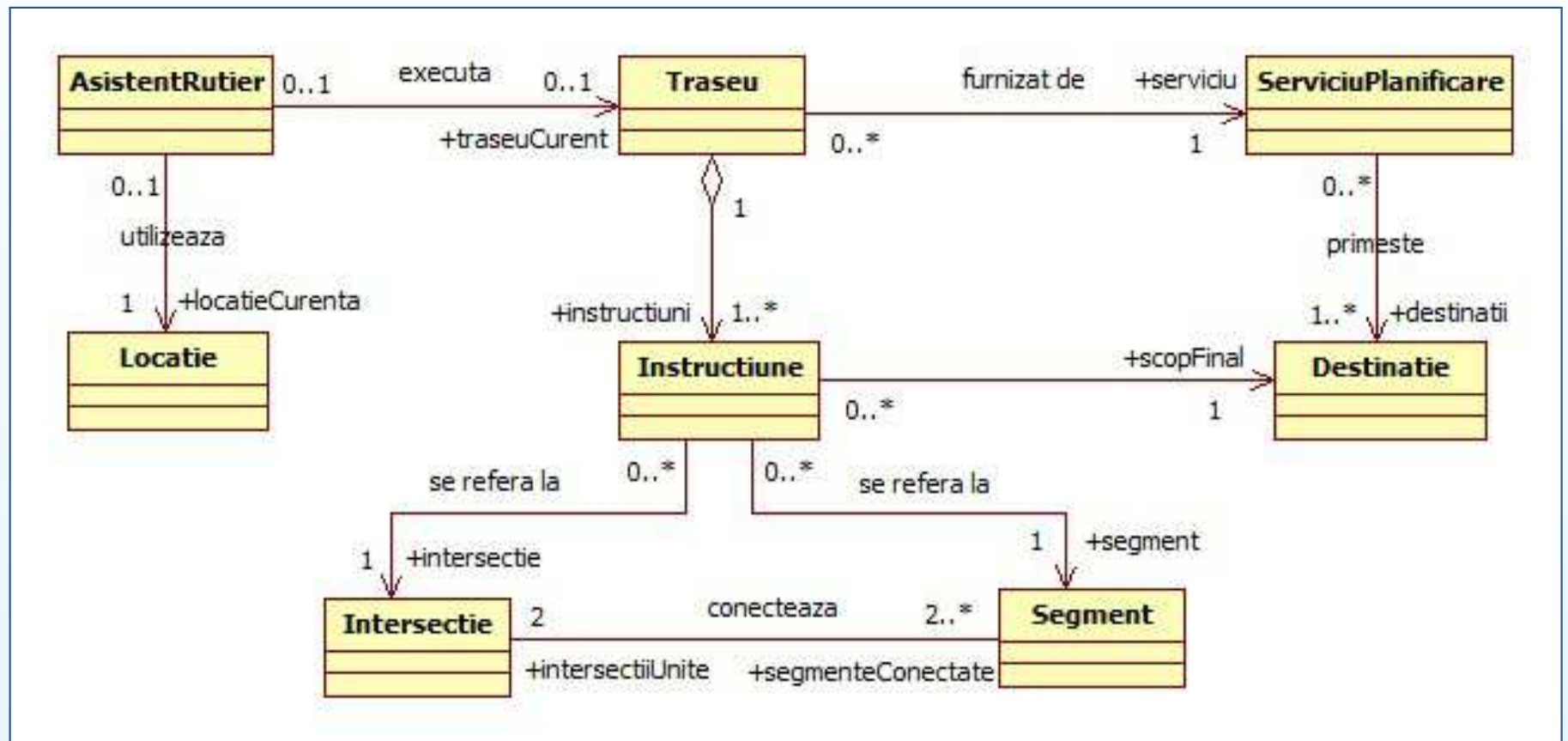
MyTrip: cerințe nefuncționale

1. Conexiunea cu serviciul de planificare este realizată prin intermediul unui modem wireless. Se presupune că acesta funcționează corect la locația inițială.
2. Odată începută călătoria, sistemul *MyTrip* trebuie să furnizeze instrucțiuni corecte, chiar și în condițiile pierderii conexiunii cu serviciul web de planificare.
3. Sistemul *MyTrip* trebuie să minimizeze timpul de conectare, pentru a reduce costurile de operare.
4. Replanificarea este posibilă doar în condițiile în care conectarea la serviciul de planificare este posibilă.
5. Serviciul de planificare trebuie să suporte cel puțin 50 de șoferi diferiți și cel puțin 1000 de trasee diferite.

MyTrip: concepte din domeniul problemei

<i>Destinație</i>	O <i>Destinație</i> reprezintă un loc în care șoferul dorește să ajungă
<i>Intersecție</i>	O <i>Intersecție</i> reprezintă un punct geografic în care se întâlnesc mai multe <i>Segmente</i>
<i>Segment</i>	Un <i>Segment</i> reprezintă calea dintre două <i>Intersecții</i>
<i>Instrucțiune</i>	Dată fiind o <i>Intersecție</i> și un <i>Segment</i> adiacent, o <i>Instrucțiune</i> descrie, în limbaj natural, modalitatea de a dirija mașina pe respectivul <i>Segment</i>
<i>Traseu</i>	Un <i>Traseu</i> constă dintr-o succesiune de <i>Instrucțiuni</i> între două <i>Destinații</i>
<i>ServiciuPlanificare</i>	Un <i>ServiciuPlanificare</i> este un serviciu Web care poate construi un <i>Traseu</i> pe baza unui șir de destinații, sub forma unei secvențe de <i>Intersecții</i> și <i>Segmente</i> asociate
<i>AsistentRutier</i>	Un <i>AsistentRutier</i> oferă <i>Instrucțiuni</i> șoferului, pe baza <i>Locației</i> curente și a următoarei <i>Intersecții</i>
<i>Locație</i>	O <i>Locație</i> reprezintă o poziție a mașinii indicată de sistemul GPS încorporat

MyTrip: modelul conceptual



Identificarea obiectivelor de proiectare

- *Obiective de proiectare* = criterii de calitate pe care sistemul trebuie să se focuseze
- Trebuie specificate explicit, pentru ca fiecare decizie luată să se conformeze aceluiași set uniform de criterii
- Categoriile de obiective de proiectare
 1. *performanță*: timp de răspuns, puterea de calcul, memorie utilizată
 2. *dependabilitate*: robustețe, fiabilitate, disponibilitate, toleranță la erori, securitate, siguranță
 3. *cost*: costuri de dezvoltare, instalare, întreținere, administrare
 4. *întreținere*: extensibilitate, modificabilitate, adaptabilitate, portabilitate, lizibilitate, trasabilitatea cerințelor
 5. *criterii utilizator*: utilitate, utilizabilitate
- Categoriile 1,2,5 pot fi deduse din cerințele nefuncționale și domeniul problemei, celelalte sunt dictate de client și furnizor
- Priorități și compromisuri: viteză vs. memorie, deadline vs. funcționalitate/calitate/personal

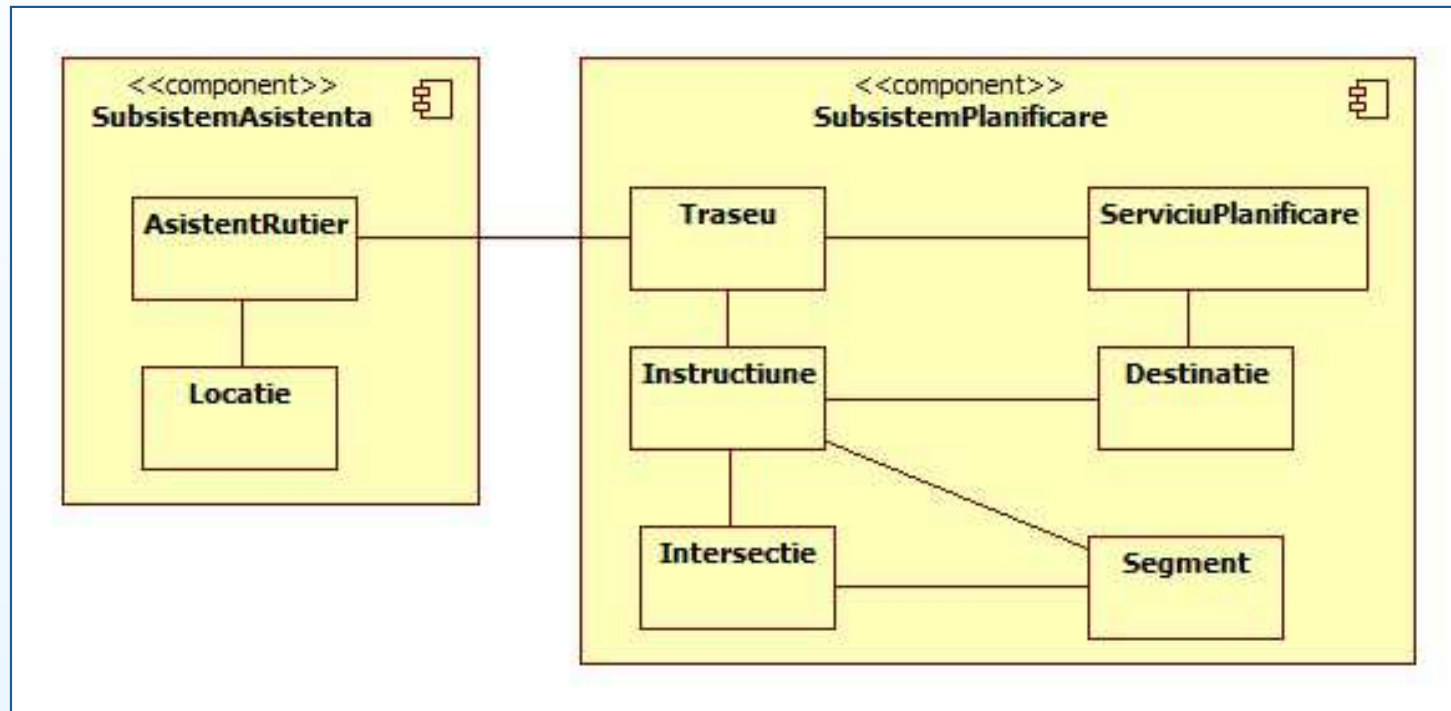
MyTrip: Obiective de proiectare

- *Fiabilitate*: Sistemul *MyTrip* trebuie să fie fiabil [generalizare a cerinței nefuncționale 2]
- *Toleranță la erori*: Sistemul *MyTrip* trebuie să funcționeze și în condițiile pierderii conexiunii între sistemul de asistență și serviciul web [reformulare a cerinței funcționale 2]
- *Securitate*: Sistemul *MyTrip* nu trebuie să permită accesul neautorizat la traseele salvate de un șofer [dedusă din domeniul problemei]
- *Modificabilitate*: Sistemul *MyTrip* trebuie să poată fi modificat pentru a utiliza un alt serviciu de planificare [anticipare a schimbării de către dezvoltatori]

Descompunerea inițială a sistemului

- Se realizează pe baza cerințelor funcționale, folosind stiluri arhitecturale predefinite
- Sistemul *MyTrip* - două grupuri de obiecte
 - cele implicate în cazul de utilizare *PlanificareTraseu*
 - cele implicate în cazul de utilizare *ExecutareTraseu*
 - Clasele *Traseu*, *Destinație*, *Instrucțiune*, *Segment*, *Intersecție* (strâns cuplate, folosite pentru a reprezenta un traseu) sunt partajate între cele două cazuri de utilizare
- Descompunere *MyTrip* - stil arhitectural *Repository*, subsistemul de planificare este responsabil de crearea structurii de date centralizate
 - Subsistemul *SubsistemPlanificare* = *ServiciuPlanificare* + clasele folosite pentru reprezentarea traseului
 - Subsistemul *SubsistemAsistenta* = restul claselor
 - O singură dependență între cele două subsisteme

MyTrip: Subsisteme și responsabilități inițiale



SubsistemPlanificare

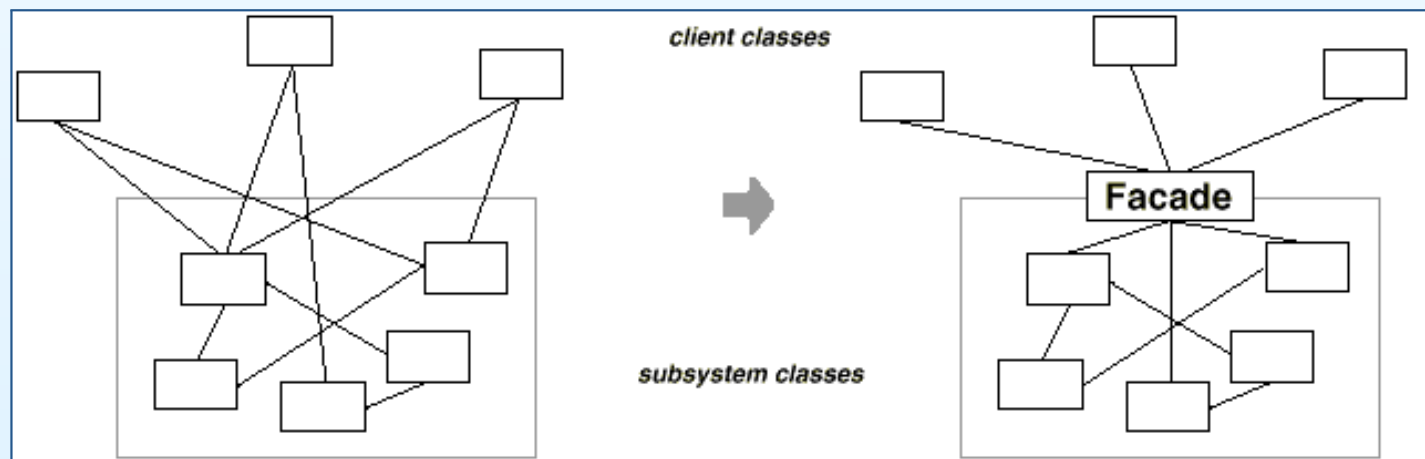
Este responsabil de construirea unui *Traseu* ce unește o serie de *Destinații*. Răspunde la cereri de replanificare din partea *SubsistemuluiAsistenta*.

SubsistemAsistenta

Este responsabil de descărcarea unui *Traseu* din *ServiciulPlanificare* și de executarea acestuia, prin furnizarea de *Instrucțiuni* șoferului, pe baza *Locației* curente.

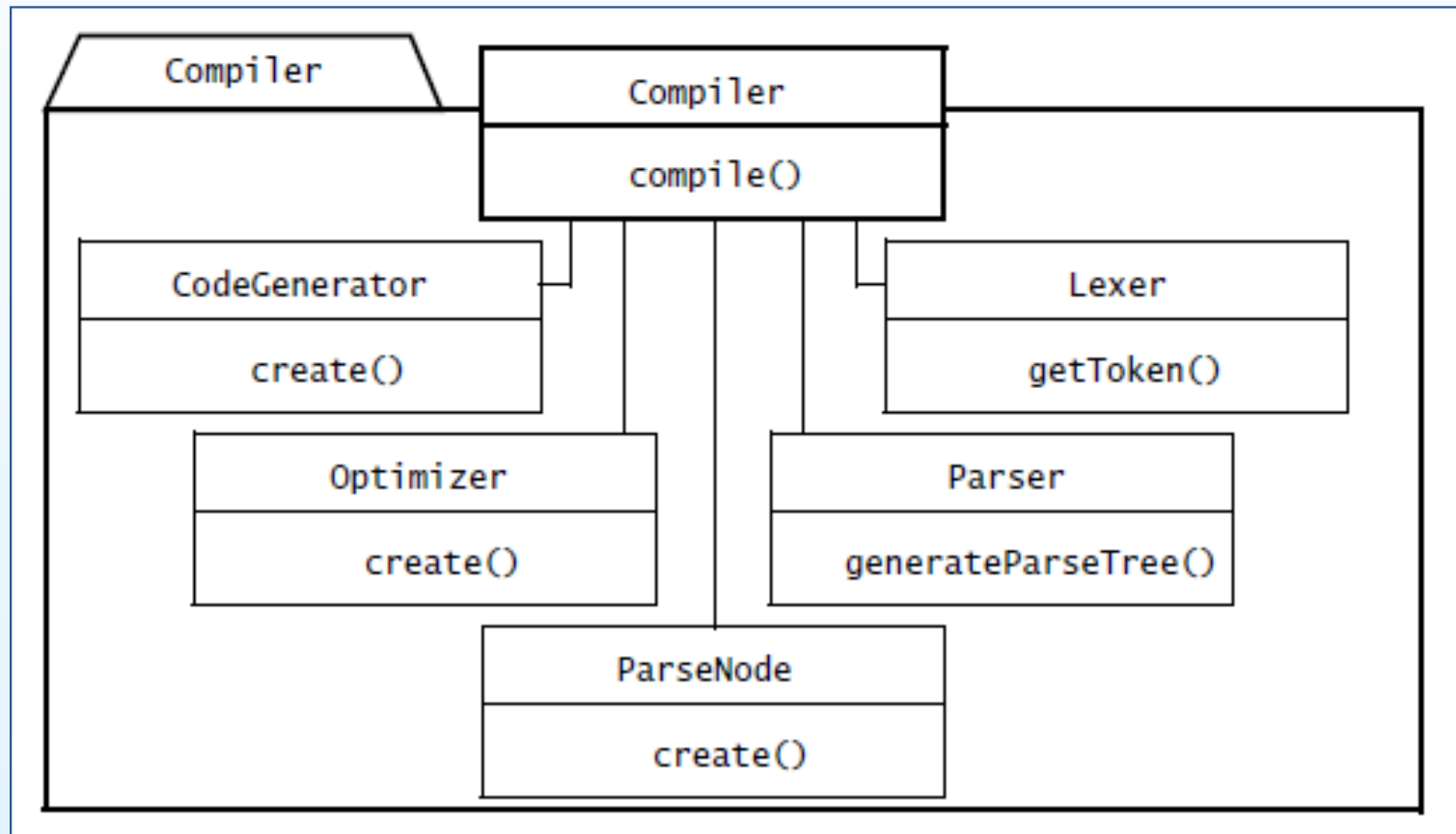
Descompunerea inițială a sistemului (cont.)

- Euristici pentru descompunerea inițială a sistemului în subsisteme
 - Asignează clasele identificate la nivelul unui caz de utilizare aceluiași subsistem
 - Creează un subsistem dedicat pentru clasele utilizate pentru transportul datelor între subsisteme, sau asignează acele clase subsistemelor responsabile de crearea lor
 - Toate clasele dintr-un subsistem trebuie să fie înrudite funcțional
 - Minimizează numărul de asocieri ce depășesc granițele subsistemelor
- Șablonul de proiectare *Facade* (structural) [Gamma et al., 1994]
 - Permite reducerea dependențelor (cuplării) dintre un subsistem și clienții săi



Exemplu de aplicare a șablonului *Facade*

- Clasa *Compiler* reprezintă o fațadă ce ascunde clasele *Lexer*, *Parser*, *ParseNode*, *Optimizer*, *CodeGenerator*



Șablonul *Facade*

- *Definiție*

- Oferă o interfață unificată, de nivel înalt, pentru un grup de interfețe ale unui subsistem, care facilitează utilizarea acestuia

- *Aplicabilitate*

- Oferirea unei interfețe simple către un subsistem complex
- Diminuarea numărului de dependențe între clienți și clasele de implementare ale unei abstractizări
- Stratificarea unui subsistem

- *Participanți*

- *Facade (Compiler)*
 - Știe ce clase ale subsistemului sunt responsabile de o cerere
 - Delegă cererile clientului către obiectele corespunzătoare din subsistem
- Clasele subistemului (*Lexer, Parser, ...*)
 - Implementează funcționalitatea subsistemului
 - Rezolvă sarcinile atribuite de obiectul fațadă
 - Nu știu de existența fațadei (nu păstrează referințe spre ea)

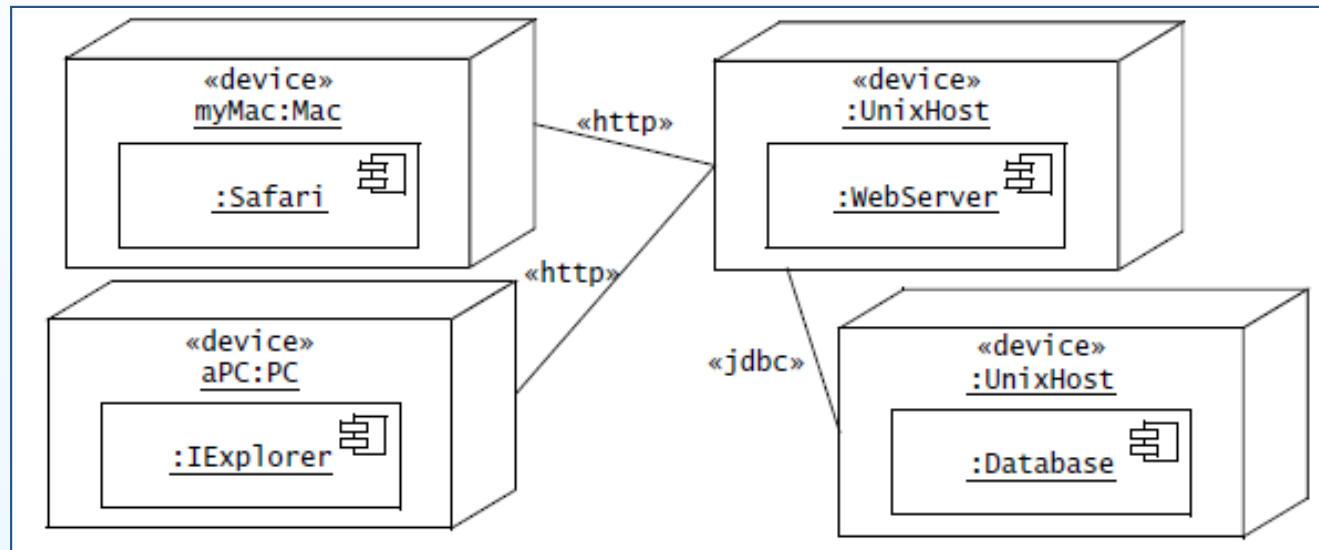
Șablonul *Facade* (cont.)

- *Colaborări*
 - Clienții comunică cu subsistemul trimițând cereri către obiectul fațadă, care le transmite către obiectele corespunzătoare din subsistem. Deși obiectele din subsistem efectuează munca reală, fațada ar putea fi obligată să efectueze sarcini proprii, pentru a translați interfața sa în interfețele subsistemului
 - Clienții care utilizează fațada nu trebuie să acceseze direct obiectele din subsistemul acesteia
- Șablonul nu interzice clienților să utilizeze direct clasele subsistemului, dacă acest lucru este necesar

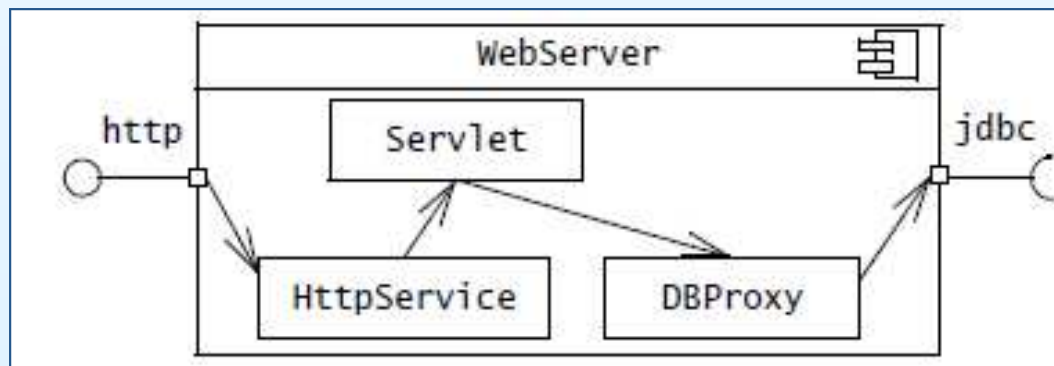
Diagrame de repartiție a resurselor

- La runtime, un sistem este reprezentat de o mulțime de componente care interacționează, distribuite la nivelul unor noduri
- O *diagramă de repartiție UML* (eng. *UML deployment diagram*) ilustrează relația dintre componentele runtime și noduri
 - *Componentă* = unitate autoconținută, care oferă servicii altor componente sau actorilor
 - Un server web este o componentă care oferă servicii browserelor web
 - Un browser web este o componentă care oferă servicii utilizatorilor
 - *Nod* = dispozitiv fizic (calculator) sau mediu de execuție pentru componente
 - Un nod poate conține un alt nod (un dispozitiv conține un mediu de execuție)
- Reprezentare
 - Nodurile sunt reprezentate ca și paralelipiede dreptunghice, ce conțin componente
 - Nodurile pot fi stereotipizate pentru a indica un dispozitiv fizic sau un mediu de execuție
 - Comunicarea dintre noduri se reprezintă cu o linie continuă, ce poate fi stereotipizată cu un nume de protocol

Exemplu de diagramă UML de repartiție

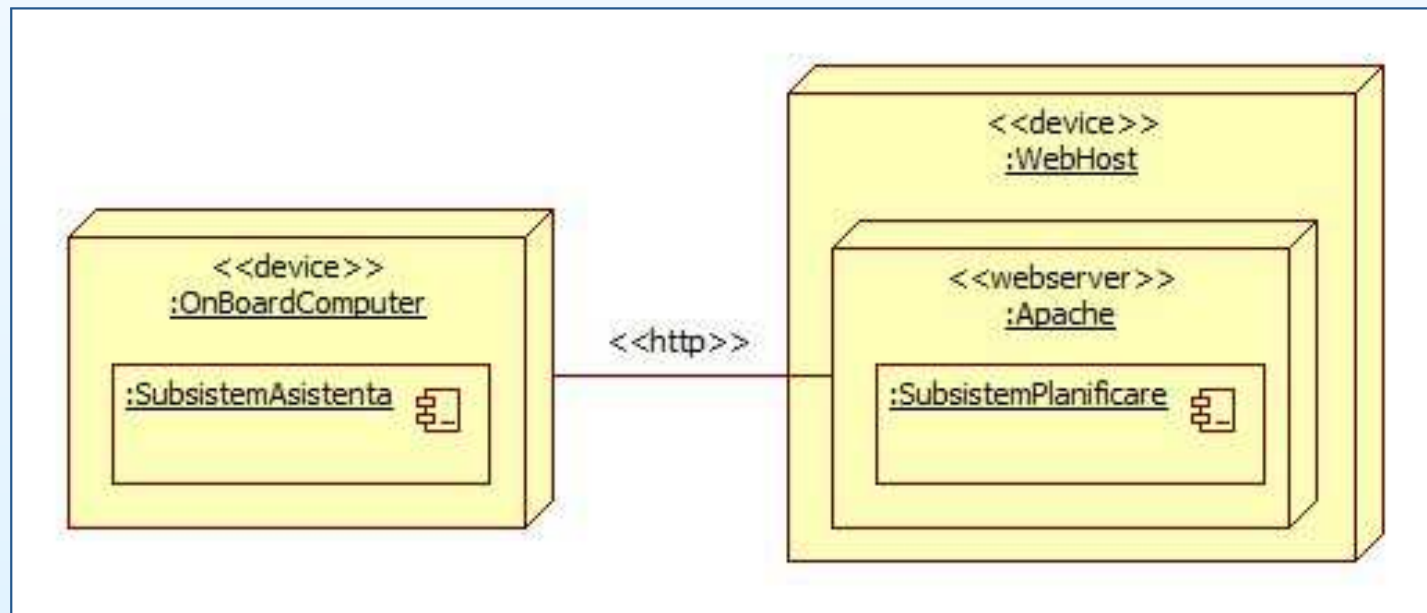


- Componentele pot fi rafinate, pentru a include informații relativ la clasele pe care le conțin și la interfețele pe care le oferă/solicită



Maparea subsistemelor la hardware

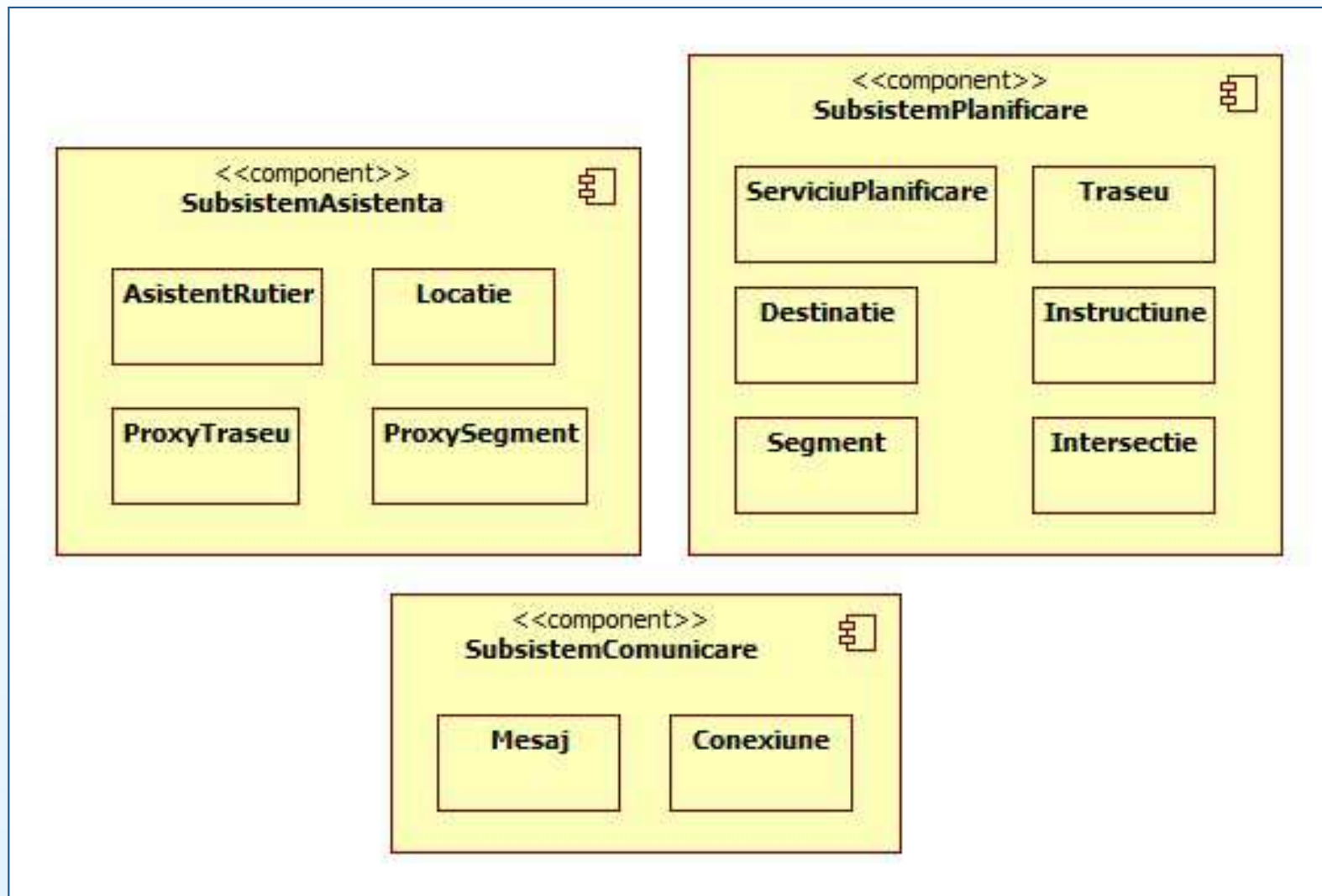
- Strategia de alocare a subsistemelor pe echipamentele hardware și proiectarea infrastructurii de comunicare între aceste subsisteme influențează semnificativ performanțele sistemului și complexitatea acestuia
 - Selectarea configurației hardware include și selectarea mașinii virtuale pe care va fi construit sistemul
- Alocarea subsistemelor *MyTrip*



Maparea subsistemelor la hardware (cont.)

- Alocarea subsistemelor *MyTrip*
 - Subsistemul *SubsistemPlanificare* rulează pe un server web (*WebHost*), în mediul de execuție Apache. Mașina virtuală e o mașină Unix
 - Subsistemul *SubsistemAsistență* rulează pe computerul de bord al mașinii (*OnBoardComputer*), mașina virtuală fiind un browser web
- Alocarea subsistemelor pe noduri, implică, de obicei, identificarea unor noi clase/subsisteme utilizate pentru transportul datelor între aceste noduri
 - În cazul *MyTrip*, obiectele aferente unui traseu (*Traseu*, *Instructiune*, *Intersecție*, *Segment*, *Destinație*) trebuie transportate de la subsistemul de planificare spre cel de asistență
 - Pentru a gestiona comunicarea între aceste două subsisteme, se introduce un subsistem nou *SubsistemComunicare*, ce va fi localizat pe ambele noduri

MyTrip: prima rafinare a descompunerii inițiale

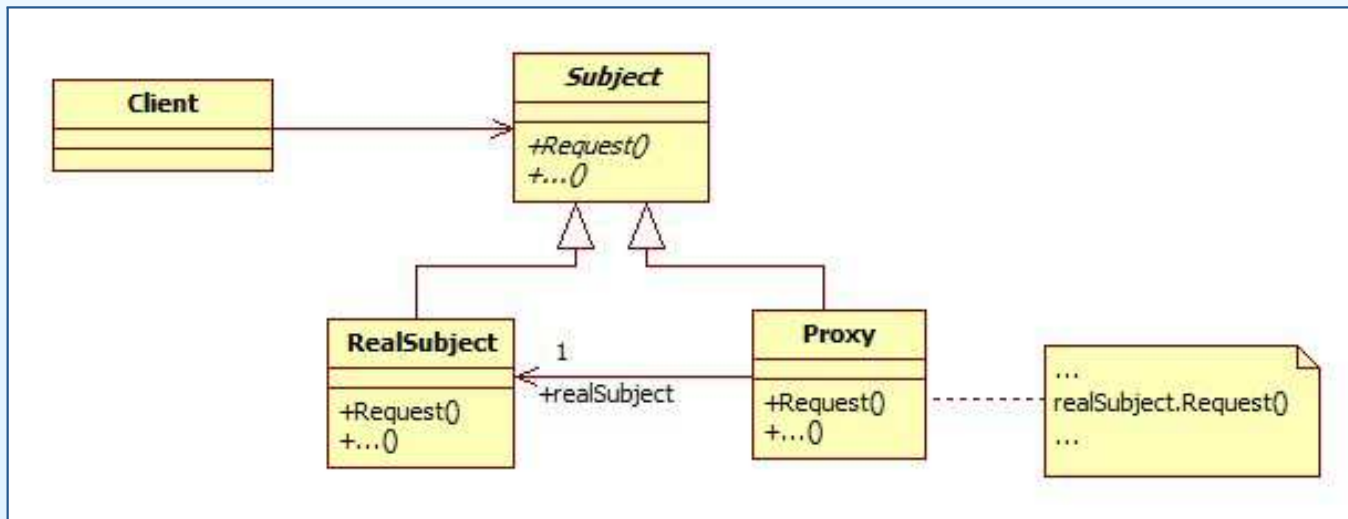


MyTrip: prima rafinare a descompunerii inițiale (cont)

<i>SubsistemComunicare</i>	Responsabil de transportarea obiectelor de la <i>SubsistemulPlanificare</i> la <i>SubsistemulAsistenta</i>
<i>Conexiune</i>	O <i>Conexiune</i> reprezintă o legătură activă între subsistemul de planificare și cel de asistență. Un obiect <i>Conexiune</i> tratează cazurile excepționale cauzate de pierderea conexiunii în rețea
<i>Mesaj</i>	Un <i>Mesaj</i> reprezintă un <i>Traseu</i> și elementele asociate (<i>Instructiune</i> , <i>Destinatie</i> , <i>Segment</i> , <i>intersecție</i>), codificate pentru transport

Șablonul *Proxy* (structural) [Gamma et al., 1994]

- *Definiție*
 - Șablonul asigură, pentru un obiect, un surogat sau un înlocuitor, în scopul controlării accesului la acesta
- *Aplicabilitate*
 - *Proxy la distanță* - oferă un reprezentant local pentru un obiect dintr-un spațiu de adresă diferit
 - *Proxy virtual* - creează la cerere obiecte costisitoare
 - ...
- *Structură*



Șablonul *Proxy* (cont.)

- *Participanți*

- *Proxy*

- Păstrează o referință care îi permite să acceseze subiectul real
- Asigură o interfață identică celei a clasei *Subject*, astfel încât un obiect proxy să poată înlocui un obiect real
- Controlează accesul la subiectul real și poate răspunde de crearea și ștergerea acestuia
- Obiectele *proxy la distanță* răspund de codificarea unei cereri și a argumentelor acesteia și de transmiterea cererii codificate către subiectul real, aflat într-un spațiu de adresă diferit
- Obiectele *proxy virtuale* pot depozita informație suplimentară despre subiectul real, astfel încât să poată amâna accesarea acestuia (ex.: un obiect *ImageProxy* poate stoca dimensiunile unui obiect *RealImage*)

- *Subject*

- Definește interfața comună pentru obiectele *RealSubject* și *Proxy*, astfel încât un obiect *Proxy* să poată fi utilizat în orice loc în care este așteptat un obiect *RealSubject*

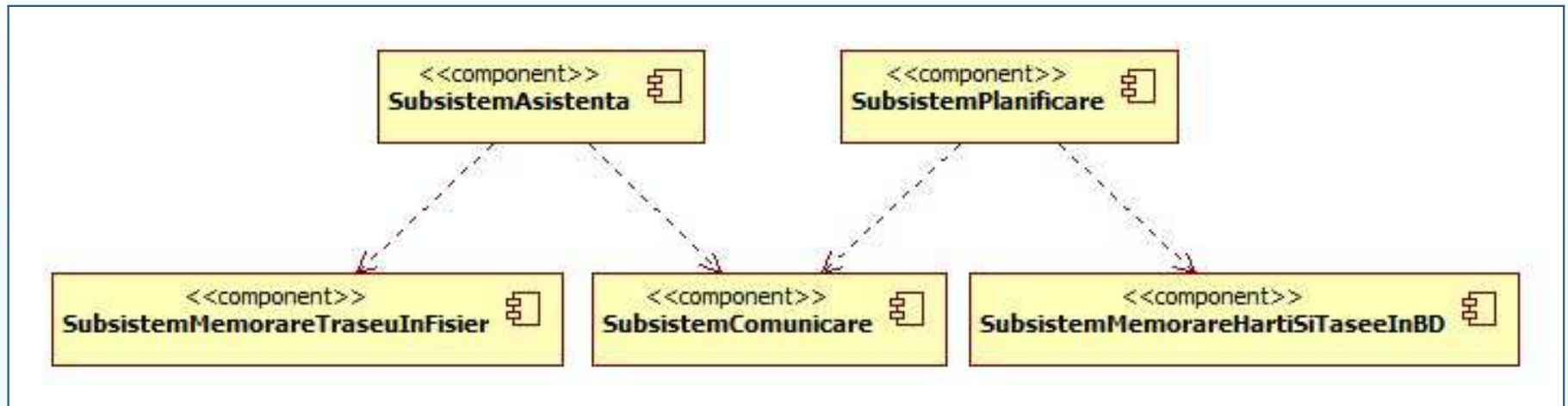
Șablonul *Proxy* (cont.)

- *RealSubject*
 - Definește obiectul real reprezentat de obiectul proxy
- *Colaborări*
 - Când se impune, obiectul proxy transmite cererile către subiectul real, funcție de tipul de proxy
- *Consecințe*
 - Șablonul *Proxy* introduce un grad de "ocolire" la accesul unui obiect. Semnificația acestei ocoliri depinde de tipul de proxy: un proxy la distanță ascunde faptul că un obiect se află într-un alt spațiu de adresă; un proxy virtual poate efectua optimizări, cum ar fi crearea unui obiect la cerere (un proxy pentru o imagine instanțiază obiectul imagine concret doar atunci când este necesară afișarea imaginii)

Gestiunea datelor persistente

- Unele dintre obiectele ce compun sistemul trebuie să fie *persistente*
 - Valorile atributelor acestora au o durată de viață ce o depășește pe cea a unei singure execuții a sistemului
- Persistența obiectelor se poate realiza folosind
 - Fișiere - în cazul în care există mai mulți cititori, dar un singur scriitor
 - Baze de date - în cazul în care datele sunt accesate de cititori și scriitori concurenți
- Strategia de persistență pentru sistemul *MyTrip*
 - Traseul curent este salvat într-un fișier, pentru a permite recuperarea acestuia în cazul în care șoferul oprește mașina înainte de a ajunge la destinația finală
 - Toate traseele aferente subsistemului de planificare, precum și hărțile necesare generării acestora sunt memorate la nivelul unei baze de date
 - Strategia de persistență aleasă determină introducerea în sistem a două noi subsisteme

MyTrip: a doua rafinare a descompunerii inițiale



SubsistemMemorareTraseuInFisier

Responsabil de memorarea traseelor în fișiere pe calculatorul de bord. Deoarece această funcționalitate este folosită doar pentru a salva trasee atunci când mașina se oprește, acest subsistem suportă doar memorarea și încărcarea rapidă a unor trasee complete.

SubsistemMemorareHartiSiTraseeInBD

Responsabil de memorarea hărților și a traseelor într-o bază de date pentru subsistemul de planificare. Acest subsistem suportă accesul concurent al mai multor șoferi și agenți de planificare.

Gestiunea datelor persistente: activități

- *Identificarea obiectelor persistente*

- Date candidat pentru persistență
 - Entitățile (nu neaparat toate) identificate în etapa de analiză (pentru sistemul *MyTrip*: *Destinație*, *Intersecție*, *Segment*, *Traseu*; *Locație* și *Instrucțiune* se recalculează funcție de poziția curentă a mașinii => nu trebuie persistate)
 - Informații legate de utilizatori, într-un sistem multi-utilizator (*Șoferi*, spre exemplu), attribute ale unor obiecte *boundary* ce rețin preferințe ale utilizatorilor, etc.
 - În general, toate clasele ale căror obiecte trebuie să supraviețuiască unei opriri a sistemului, controlată sau neașteptată

- *Selectarea strategiei de memorare*

- Decizie complexă, determinată, în general, de cerințe nefuncționale:
Obiectele trebuie regăsite rapid? Sunt necesare interogări complexe pentru regăsirea acelor obiecte? Obiectele necesită mult spațiu pentru memorare?
- În general, 3 opțiuni

Gestiunea datelor persistente: activități (cont.)

- *Fișiere*
 - + Avantaje: permit o mare varietate de optimizări de viteză și dimensiune
 - Dezavantaje: lasă pe umerii aplicației gestiunea accesului concurrent sau recuperarea datelor în urma unei căderi a sistemului
- *Baze de date relaționale*
 - + Avantaje: tehnologie matură, ce oferă servicii pentru controlul accesului, gestiunea concurenței și recuperarea datelor în urma unei căderi a sistemului
 - Dezavantaje: deși scalabile și ideale pentru volume mari de date, sunt relativ lente pentru seturi de date mici sau date nestructurate (imagini, text în limbaj natural); în plus, există necesitatea mapării obiect-relaționale
- *Baze de date orientate obiect*
 - + Avantaje: un nivel mai înalt de abstractizare: memorează datele ca și obiecte, înlăturând necesitatea translatării între obiecte și entitățile memorate
 - Dezavantaje: mai lente decât bazele de date relaționale

Euristici de selectare a mecanismului de persistență

- Când se folosesc fișiere?
 - Date voluminoase (imagini)
 - Date temporare
 - Densitate mică a informației (log-uri)
- Când se folosesc baze de date?
 - Acces concurent
 - Mai multe platforme sau aplicații care accesează aceleași date
- Când se folosesc baze de date relaționale?
 - Interogări complexe
 - Seturi mari de date
- Când se folosesc baze de date orientate obiect?
 - Utilizare masivă a asocierilor pentru regăsirea datelor
 - Seturi de date medii

Definirea politicilor privind controlul accesului

- În sistemele multi-utilizator, actori diferiți au, de obicei, drepturi de acces diferite la diferite funcționalități și date
- Cum modelăm accesul?
 - În timpul analizei - prin asocierea diferitor cazuri de utilizare la diferiți actori
 - În timpul proiectării - prin determinarea acelor obiecte partajate între actori, precum și a modului în care actorii pot controla accesul. Funcție de cerințele de securitate ale sistemului, definim și modul în care actorii se autentifică în sistem, precum și modalitatea de criptare a anumitor date
- Sistemul *MyTrip*
 - Se introduce o clasă *Șofer*, asociată clasei *Traseu*, pentru a asigura trimiterea traseelor doar la șoferii care le-au creat
 - Subsistemul de planificare devine responsabil de autentificarea șoferilor, înainte de a le trimite traseele solicitate
 - Subsistemul de comunicare criptează traficul dintre subsistemul de asistență și cel de planificare

MyTrip - o nouă rafinare

Șofer

Un Șofer reprezintă un utilizator autentificat. Este utilizat de către subsistemul de comunicare, pentru a memora cheia utilizată pentru criptare și de către subsistemul de planificare, pentru a asocia Traseele cu utilizatorii.

SubsistemPlanificare

Este responsabil de construirea unui Traseu ce unește o serie de Destinații. Răspunde la cereri de replanificare din partea SubsistemuluiAsistentă. Înainte de a procesa o cerere, subsistemul de planificare autentifică Șoferul de la subsistemul de asistență. Șoferul autentificat este utilizat pentru a determina acele Trasee care pot fi trimise subsistemului de asistență

SubsistemComunicare

Responsabil de transferul obiectelor de la SubsistemulPlanificare la SubsistemulAsistentă. Acesta utilizează Șoferul asociat Traseului care se transferă pentru a selecta o cheie și a cripta traficul.

Matrici de acces

- Accesul la clase poate fi modelat folosind *matrici de acces*
 - Liniile reprezintă actorii din sistem
 - Coloanele reprezintă clasele ale căror drepturi de acces dorim să fie modelate
 - O intrare în matrice listează operațiile care pot fi executate pe o instanță a clasei de pe coloană de către actorul de pe linie
- Variații de implementare
 - *Tabel de acces global* - reprezintă explicit fiecare triplet (actor, clasă, operație)
 - *Liste de control a accesului* - asociază o listă de perechi (actor, operație) fiecărei clase. De fiecare dată când o instanță a clasei este accesată, se verifică existența perechii (actor, operație) în listă
 - *Capabilități* - asociază perechi (clasă, operație) unui actor

Stabilirea fluxului global de control

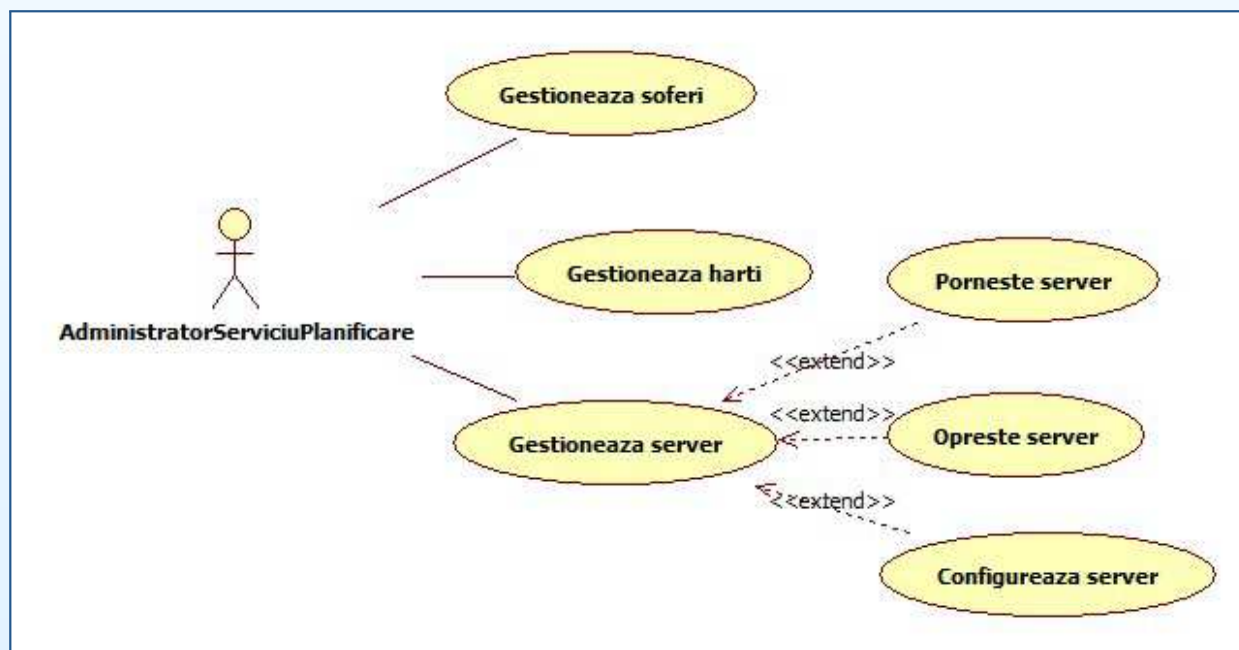
- *Flux de control* = secvențierea acțiunilor într-un sistem (ce operații se execută și în ce ordine)
- Mecanisme
 - *Flux de control procedural*
 - Operațiile așteaptă input ori de câte ori au nevoie de date de la un actor
 - Utilizat mai mult în limbajele procedurale
 - Util pentru testarea subsistemelor
 - *Flux de control dirijat de evenimente*
 - O buclă așteaptă evenimente externe. Ori de câte ori un astfel de eveniment apare, este direcționat către obiectul aferent, pe baza unor informații asociate evenimentului
 - Avantaje: abordare matură, structură simplă, centralizarea input-urilor
 - *Thread-uri*
 - Varianta concurentă a controlului procedural
 - Sistemul poate crea un număr arbitrar de thread-uri, fiecare răspunzând la un eveniment diferit. Dacă un thread are nevoie de date suplimentare, așteaptă input de la un actor
 - Avantaje: mecanism intuitiv; dezavantaje: dificil de testat și depanat

Descrierea cazurilor limită

- Examinarea condițiilor limită
 - Cum este sistemul pornit, inițializat și oprit?
 - Cum sunt gestionate datele corupte sau căderile de rețea?
 - Cazurile de utilizare care gestionează aceste condiții se numesc *cazuri de utilizare limită* (eng. *boundary use cases*)
- Sistemul *MyTrip*
 - Cum sunt încărcate hărțile în serviciul de planificare?
 - Cum este instalat sistemul într-o mașină?
 - De unde cunoaște sistemul *MyTrip* serviciul de planificare la care trebuie să se conecteze?
 - Cum sunt adăugați șoferii în serviciul de planificare?
- Euristici pentru determinarea cazurilor limită
 - *Configurare*: Pentru fiecare obiect persistent, se identifică acele cazuri de utilizare în care obiectul este creat/ distrus. Pentru obiectele care nu sunt create/distruse în cazurile de utilizare existente (ex.: hărțile în sistemul *MyTrip*), se adaugă un caz de utilizare gestionat de un administrator al sistemului (ex.: *GestionareHarti*)

Descrierea cazurilor limită (cont.)

- *Pornire/oprire*: Pentru fiecare componentă, se adaugă trei cazuri de utilizare pentru a porni, opri și configura componenta
 - *Gestiunea excepțiilor*: Pentru fiecare tip de eșec (ex.: cădere de rețea), se creează un caz de utilizare excepțional, care extinde un caz de utilizare existent
- Cazuri de utilizare limită pentru *MyTrip*



Referințe

- [Gamma et al., 1994] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA, 1994.